AMENDMENT TO THE CLAIMS

Claims 1-2, 4-22, 33-34, and 36-60 are currently pending in the Application. Claims 1-2, 4, 6-10, 13-14, 16, 21, 33-34, 36, 38-42, 46, and 53-55 are currently amended to clarify what Applicants regard as the claimed subject matter(s) as embodied in these claims, without acquiescence in the cited basis for rejection or prejudice to pursue the original claims in a related application. Claims 61-64 are new. A complete listing of the current pending claims is provided below and supersedes all previous claims listing(s). No new matter has been added.

1. (Currently Amended) A method for performing performance analysis for a target machine which comprises a software portion and a hardware portion, comprising:

describing a design for the target machine as a network of logical entities;

selecting at least one of the logical entities for a software implementation;

implementing a source software program for the logical entities selected for the software implementation;

generating an optimized assembler code for the software program, wherein the optimized assembler code is an assembly-language representation of the software implementation;

performing a performance analysis using the optimized assembler code, wherein the act of performing the performance analysis is performed by a processor;

generating a software simulation model in a high level language format based at least in part upon the optimized assembler code by disassembling a binary code ~~into a high-level language format~~ and by annotating the software simulation model with information related to hardware on which the software ~~program~~ implementation runs based at least in part upon a result of the act of performing the performance analysis to capture a dynamic interaction between tasks during runtime, wherein the act of annotating the software simulation model is performed during a time [[when]]of the act of generating [[a]]the software simulation model ~~by translating the assembler code or disassembling a binary code~~;

storing the software simulation model on a computer usable storage medium;

generating a hardware and software co-simulation model using the software simulation model; and

storing at least the hardware and software co-simulation model on the computer usable storage medium or a second computer usable storage medium or displaying the at least the hardware and software co-simulation model on a display apparatus.

2. (Currently Amended) The method of claim 1, wherein the ~~compiling step~~ act of generating the software simulation model further comprises incorporating a description of the target machine.

3. (Cancelled).

4. (Currently Amended) The method of claim 1, further comprising selecting at least one of the network of logical entities for a hardware implementation, and using an existing software model of the hardware implementation from the ~~selected~~ at least one of the network of logical entities, wherein the hardware and software co-simulation model is generated using the existing software model of the hardware implementation.

5. (Original) The method of claim 1, wherein the performance analysis measures an execution time of an element of the assembler code.

6. (Currently Amended) The method of claim 1, wherein the software program is compiled using [[the]]a same compiler used to compile a production executable.

7. (Currently Amended) The method of claim 1, wherein the act of performing the performance analysis comprises annotating the optimized assembler code with performance information.

8. (Currently Amended) The method of claim 7, wherein the performance information [[is]]comprises timing information.

9. (Currently Amended) A method of preparing software for a performance estimation, comprising:

obtaining a software assembly code module from a source code module, wherein the software assembly code module is an assembly-language representation;

generating a software simulation model in a high level language format by disassembling a binary code ~~into the software simulation model in a high level language format~~, wherein the software assembly code module comprises the binary code, and the act of generating the software simulation model is performed by a processor;

annotating the software simulation model with performance information of [[a]]hardware together with which the software simulation model runs to capture a dynamic interaction between tasks during runtime, wherein the act of annotating the software simulation model is performed during a time [[when]]of the act of generating [[a]]the software simulation model ~~by disassembling a binary code into the software simulation model~~; and

storing at least the software simulation model on a computer usable storage medium or displaying the at least the software simulation model on a display apparatus,

wherein the software simulation model is an assembler-level software simulation model, expressed in a high-level programming language.

10. (Currently Amended) The method of claim 9, wherein the act of obtaining the ~~providing a~~ software assembly code module comprises compiling software source code to assembly.

11. (Previously Presented) The method of claim 10, wherein the software assembly code module is compiled using a compiler adapted to create code that will execute on a first machine architecture.

12. (Previously Presented) The method of claim 11, wherein the performance information is associated with the first machine architecture.

13. (Currently Amended) The method of claim 11, wherein the software simulation model is compiled to execute on a second machine architecture, the second machine architecture being different from the first machine architecture.

14. (Currently Amended) The method of claim 1, wherein generating [[an]]the optimized assembler code comprises disassembling software binary code to assembly code.

15. (Previously Presented) The method of claim 9, wherein the high-level programming language comprises a C code programming language.

16. (Currently Amended) The method of claim 9, wherein the ~~translation step~~ act of generating the software simulation model further comprises gathering information from the source code module from which the software assembly code module was obtained.

17. (Previously Presented) The method of claim 16, wherein the information gathered comprises high-level hints about the software assembly code module.

18. (Previously Presented) The method of claim 9, wherein the performance information comprises estimated performance information.

19. (Previously Presented) The method of claim 9, wherein the performance information is statically estimated.

20. (Previously Presented) The method of claim 9, wherein the performance information is dynamically computed at run-time, using a formula provided during the annotating step.

21. (Currently Amended) The method of claim 9, further comprising:

compiling the software simulation model to a simulator host program; and

executing the simulator host program on a simulator to allow one or more performance measurements to be taken.

22. (Previously Presented) The method of claim 21, further comprising linking an already-annotated module with the simulation model.

23-32. (Cancelled).

33. (Currently Amended) A computer program product that includes a ~~tangible volatile or non-volatile~~ computer usable storage medium ~~useable by a processor~~, the computer usable storage medium comprising a sequence of instructions which, when executed by said processor, causes said processor to execute a ~~method~~ process for performing software performance analysis for a target machine, the process comprising:

describing a system design as a network of logical entities;

selecting at least one of the logical entities for a software implementation;

implementing a source software program for the logical entities selected for the software implementation;

~~compiling the software program to generate~~ generating an optimized assembler code ~~representation of~~ for the software program, wherein the optimized assembler code is an assembly-language representation of the software implementation;

performing a performance analysis using the optimized assembler code, wherein the act of performing the performance analysis is performed by a processor;

generating a software simulation model in a high level language format based at least in part upon the optimized assembler code by disassembling a binary code ~~into a high level language format~~ and by annotating the software simulation model with information related to hardware on which the software implementation runs based at least in part upon a result of the act of performing the performance analysis to capture a dynamic interaction between tasks during runtime, wherein the act of annotating the software simulation model is performed during a time [[when]]of the act of generating [[a]]the software simulation model ~~by disassembling the binary code~~;

storing the software simulation model on a computer usable storage medium;

generating a hardware and software co-simulation model using the software simulation model; and

storing at least the hardware and software co-simulation model on the computer usable storage medium or a second computer usable storage medium or displaying the at least the hardware and software co-simulation model on a display apparatus.

34. (Currently Amended) The computer program product of claim 33, wherein the ~~compiling step~~ act of generating the optimized assembler code further comprises incorporating a description of the target machine.

35. (Cancelled).

36. (Currently Amended) The computer program product of claim 33, the process further comprising selecting at least one of the logical entities for a hardware implementation, and synthesizing a software model of the hardware implementation from the selected logical entities, wherein the hardware and software co-simulation model is generated using the software model of the hardware implementation.

37. (Previously Presented) The computer program product of claim 33, wherein the performance analysis measures an execution time of an element of the assembler code.

38. (Currently Amended) The computer program product of claim 33, wherein the software program is compiled using [[the]]a same compiler used to compile a production executable.

39. (Currently Amended) The computer program product of claim 33, wherein performing the performance analysis comprises annotating the optimized assembler code with performance information.

40. (Currently Amended) The computer program product of claim 39, wherein the performance information [[is]]comprises timing information.

41. (Currently Amended) A computer program product that includes a ~~tangible volatile or non-volatile~~ computer usable storage medium ~~useable by a processor~~, the medium comprising a sequence of instructions which, when executed by said processor, causes said processor to execute a method for preparing software for a performance estimation, comprising:

obtaining a software assembly code module from a source code module, wherein the software assembly code module is an assembly-language representation;

generating a simulation model in a high level language format by disassembling a binary code ~~into simulation model in a high level language format~~, wherein the software assembly code module comprises the binary code, and the act of generating the software simulation model is performed by a processor;

annotating the simulation model with performance information of hardware together with which the simulation model runs to capture a dynamic interaction between tasks during runtime, wherein the act of annotating the software simulation model is performed during a time [[when]]of the act of generating [[a]]the software simulation model ~~by disassembling a binary code into the software simulation model~~; and

storing at least the simulation model on a computer usable storage medium or displaying the at least the software simulation model on a display apparatus,

wherein the simulation model is an assembler-level software simulation model, expressed in a high-level programming language.

42. (Currently Amended) The computer program product of claim 41, wherein ~~providing a~~ obtaining the software assembly code module comprises compiling software source code to assembly.

43. (Previously Presented) The computer program product of claim 42, wherein the software assembly code module is compiled using a compiler adapted to create code that will execute on a first machine architecture.

44. (Previously Presented) The computer program product of claim 43, wherein the performance information is associated with the first machine architecture.

45. (Previously Presented) The computer program product of claim 43, wherein the simulation model is compiled to execute on a second machine architecture, the second machine architecture being different from the first machine architecture.

46. (Currently Amended) The computer program product of claim 41, wherein ~~providing a~~ obtaining the software assembly code module comprises disassembling software binary code to assembly code.

47. (Previously Presented) The computer program product of claim 41, wherein the high-level programming language comprises a C code programming language.

48. (Previously Presented) The computer program product of claim 41, wherein the translation step further comprises gathering information from the source code module from which the assembly code module was obtained.

49. (Previously Presented) The computer program product of claim 48, wherein the information gathered comprises high-level hints about the software assembly code module.

50. (Previously Presented) The computer program product of claim 41, wherein the performance information comprises estimated performance information.

51. (Previously Presented) The computer program product of claim 41, wherein the performance information is statically estimated.

52. (Previously Presented) The computer program product of claim 41, wherein the performance information is dynamically computed at run-time, using a formula provided during the annotating step.

53. (Currently Amended) The computer program product of claim 41, the process further comprising:

compiling the simulation model to a simulator host program; and

executing the simulator host program on a simulator to allow performance measurements to be taken.

54. (Currently Amended) The computer program product of claim 53, the process further comprising linking an already-annotated module with the simulation model.

55. (Currently Amended) A method of translating an assembly language software module into a simulation model, comprising:

receiving the assembly language software module;

parsing the assembly language software module into a data structure, the data structure comprising one or more nodes, each of the one or more nodes being mapped to a period of time using a mapping definition, each of the one or more nodes containing an element of the assembly language software module;

processing, by using a processor, the data structure to refine accuracy of an assembler-level software simulation model by generating the assembler-level software simulation model based on the assembly language software module by using the assembly language software module or by disassembling a binary code, wherein the assembler-level software simulation model is expressed in a high-level programming language and is used to determine a time slot;

associating performance information comprising a predicted execution delay with an element of the assembly language software module to capture a dynamic interaction between tasks during runtime, wherein the act of associating is performed during a time [[when]]of the act of parsing the assembly language software into a data structure; and

displaying a result of the simulation model on a display apparatus or storing the result of the simulation model in a ~~tangible~~ computer ~~readable~~ usable storage medium.

56. (Previously Presented) The method of claim 55, wherein the one or more nodes comprises a first node and a second node, the first node being mapped to a first period of time, the second node being mapped to a second period of time, the first period of time being different from the second period of time.

57. (Previously Presented) The method of claim 55, wherein the performance information comprises an execution delay value for the element of the assembly language software module.

58. (Previously Presented) The method of claim 55, wherein the performance information is a statically computed value.

59. (Previously Presented) The method of claim 55, wherein the performance information is a formula for dynamically computing a value.

60. (Previously Presented) The method of claim 55, wherein processing the data structure comprises replicating the behavior of the assembly language software model in the simulation model.

61. (New) A system for performing performance analysis for a target machine which comprise a software portion and a hardware portion, comprising:

means for describing a design for the target machine as a network of logical entities;

means for selecting at least one of the logical entities for a software implementation;

means for implementing a source software program for the logical entities selected for the software implementation;

means for generating an optimized assembler code for the software program, wherein the optimized assembler code is an assembly-language representation of the software implementation;

means for performing a performance analysis using the optimized assembler code, wherein the means for performing the performance analysis comprises a processor;

means for generating a software simulation model in a high level language format based at least in part upon the optimized assembler code by disassembling a binary code and by annotating the software simulation model with information related to hardware on which the software implementation runs based at least in part upon an execution result of the means for performing the performance analysis to capture a dynamic interaction between tasks during runtime, wherein the means for annotating the software simulation model is invoked during a time when the means for generating the software simulation model executes;

a computer usable storage medium configured for storing the software simulation model;

means for generating a hardware and software co-simulation model using the software simulation model; and

a second computer usable storage medium or the computer usable storage medium configured for storing at least the hardware and software co-simulation model or a display apparatus configured for displaying the at least the hardware and software co-simulation model.

62. (New) A system of preparing software for a performance estimation, comprising:

means for obtaining a software assembly code module from a source code module, wherein the software assembly code module is an assembly-language representation;

means for generating a software simulation model in a high level language format by disassembling a binary code, wherein the software assembly code module comprises the binary code, and the means for generating the software simulation model comprises a processor;

means for annotating the software simulation model with performance information of hardware together with which the software simulation model runs to capture a dynamic interaction between tasks during runtime, wherein the means for annotating the software simulation model is invoked during a time when the means for generating the software simulation model executes; and

a computer usable storage medium configured for storing at least the software simulation model on a computer usable storage medium or displaying the at least the software simulation model on a display apparatus, wherein the software simulation model is an assembler-level software simulation model, expressed in a high-level programming language.

63. (New) A system of translating an assembly language software module into a simulation model, comprising:

means for receiving the assembly language software module;

means for parsing the assembly language software module into a data structure, the data structure comprising one or more nodes, each of the one or more nodes being mapped to a period of time using a mapping definition, each of the one or more nodes containing an element of the assembly language software module;

means for processing the data structure to refine accuracy of an assembler-level software simulation model by generating the assembler-level software simulation model based on the assembly language software module by using the assembly

language software module or by disassembling a binary code, wherein the assembler-level software simulation model is expressed in a high-level programming language and is used to determine a time slot, and the means for processing the data structure comprises a processor;

means for associating performance information comprising a predicted execution delay with an element of the assembly language software module to capture a dynamic interaction between tasks during runtime, wherein the means for associating is invoked during a time when the means for parsing the assembly language software executes; and

a display apparatus configured for displaying a result of the simulation model or a computer usable storage medium configured for storing the result of the simulation model.

64. (New) A computer program product that includes a computer usable storage medium, the medium comprising a sequence of instructions which, when executed by said processor, causes said processor to execute a method for translating an assembly language software module into a simulation model, comprising:

receiving the assembly language software module;

parsing the assembly language software module into a data structure, the data structure comprising one or more nodes, each of the one or more nodes being mapped to a period of time using a mapping definition, each of the one or more nodes containing an element of the assembly language software module;

processing, by using a processor, the data structure to refine accuracy of an assembler-level software simulation model by generating the assembler-level software simulation model based on the assembly language software module by using the assembly language software module or by disassembling a binary code, wherein the assembler-level software simulation model is expressed in a high-level programming language and is used to determine a time slot;

associating performance information comprising a predicted execution delay with an element of the assembly language software module to capture a dynamic interaction between tasks during runtime, wherein the act of associating is performed during a time of the act of parsing the assembly language software into a data structure; and

displaying a result of the simulation model on a display apparatus or storing the result of the simulation model in a computer usable storage medium.